SANDIA REPORT

SAND2005-7413
Unlimited Release
Printed November 2005

SAR Polar Format Implementation with MATLAB

Grant D. Martin, Armin W. Doerry

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2005-7413
Unlimited Release
Printed November 2005

SAR Polar Format Implementation with MATLAB

Grant D. Martin
Embedded Radar Processing Department

Armin W. Doerry
SAR Applications Department

Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87185-1330

ABSTRACT

Traditional polar format image formation for Synthetic Aperture Radar (SAR) requires a large amount of processing power and memory in order to accomplish in real-time. These requirements can thus eliminate the possible usage of interpreted language environments such as MATLAB. However, with trapezoidal aperture phase history collection and changes to the traditional polar format algorithm, certain optimizations make MATLAB a possible tool for image formation. Thus, this document's purpose is two-fold. The first outlines a change to the existing Polar Format MATLAB implementation utilizing the Chirp Z-Transform that improves performance and memory usage achieving near real-time results for smaller apertures. The second is the addition of two new possible image formation options that perform a more traditional interpolation style image formation. These options allow the continued exploration of possible interpolation methods for image formation and some preliminary results comparing image quality are given.

CONTENTS

1. Introduction & Background

It is well known that SAR image formation can be accomplished via a 2-D Fourier Transform, since the data collected represents the Fourier Space of the 3-D scene being imaged [1]. However, when a large scene is imaged at a fine resolution, a simple 2-D Fourier Transform is inadequate due to target Motion Through Resolution Cells (MTRC) [2]. One technique to alleviate image smearing caused by MTRC, is known as polar format processing [3]. In general, the raw phase history data collected for the scene represents a polar raster in the Fourier domain, so polar format processing is about reformatting the data to a Cartesian grid array for efficient digital processing and image formation of the collected SAR data.

In a real-time system, the polar-to-rectangular "re-gridding" operation is the most processing intensive. Typically, 2-D data interpolation techniques are used to reformat the data, although it is possible to manipulate radar hardware to vary pulse-to-pulse waveform and sampling parameters to considerably simplify the subsequent resampling, and limit it to a 1-D operation, thereby helping reduce some of the processing time required. Sandia radar systems are capable of this and often collect data on a trapezoidal grid that is both uniformly spaced in the range direction; and per each row, uniformly spaced in the azimuth direction. This trapezoidal grid leads to a traditional polar format reconstruction algorithm with a simplified procedure that is to interpolate each row in the azimuth direction to the Cartesian grid, so that a 2-D Fast Fourier Transform (FFT) can be used to generate the image.

A second reconstruction algorithm implemented for real-time polar-format processing is accomplished through a variation on the traditional interpolation / 2-D FFT algorithm mentioned above [4]. First, note that due to separability, the total 2-D FFT operation can be broken into a 1-D azimuth FFT and a 1-D range FFT. Therefore, the traditional polar format processing method is broken up into 3 steps. First, interpolate each row in the azimuth direction; second, perform a 1-D FFT in the azimuth direction; and third, a 1-D FFT in the range direction. The variation for the second algorithm is a changed order of operations from the above method. This change is a 1-D azimuth FFT, followed by a linear resampling in the azimuth direction, and finally a 1-D range FFT. The notable change here is the resampling in the azimuth direction is performed after the azimuth FFT. This option is available due to the linear spacing provided by the azimuth sampling of the phase history data inherent with a trapezoidal aperture.

The third, alternative reconstruction algorithm for the trapezoidal grid uses the "Chirp" Z-Transform along the azimuth direction, followed by a FFT in the range direction [1]. This method produces excellent results when implemented as a fast-convolution. However, while avoiding the errors of interpolation, it is often even less computationally efficient when compared to other methods that give similar quality results.

In order to understand the MATLAB implementations of the 3 different algorithms, a general model for the collected SAR data is presented as well as an explanation of how image formation is carried out for each model. The Detailed Analysis section of this document exhibits the code as well as improvements and some preliminary results of the different methods.

Phase History Model

The phase history model for collected SAR data as derived in [4] is.

$$X_V(i,n) \approx A_R \sigma(s) \exp j\{k_x(i,n)s_x + k_y(i)s_y\} \quad (1)$$

where the Fourier samples are located at

$$k_x(i,n) = \frac{2}{c}\left[\omega_0 + \gamma_0 T_{s,0} i\right] \cdot \cos\psi_0 \cdot d\alpha \cdot n, \quad (2)$$

$$k_y(i) = -\frac{2}{c}\left[\omega_0 + \gamma_0 T_{s,0} i\right] \cdot \cos\psi_0 \quad (3)$$

This model defines a trapezoidal grid in the Fourier Space of the scene.

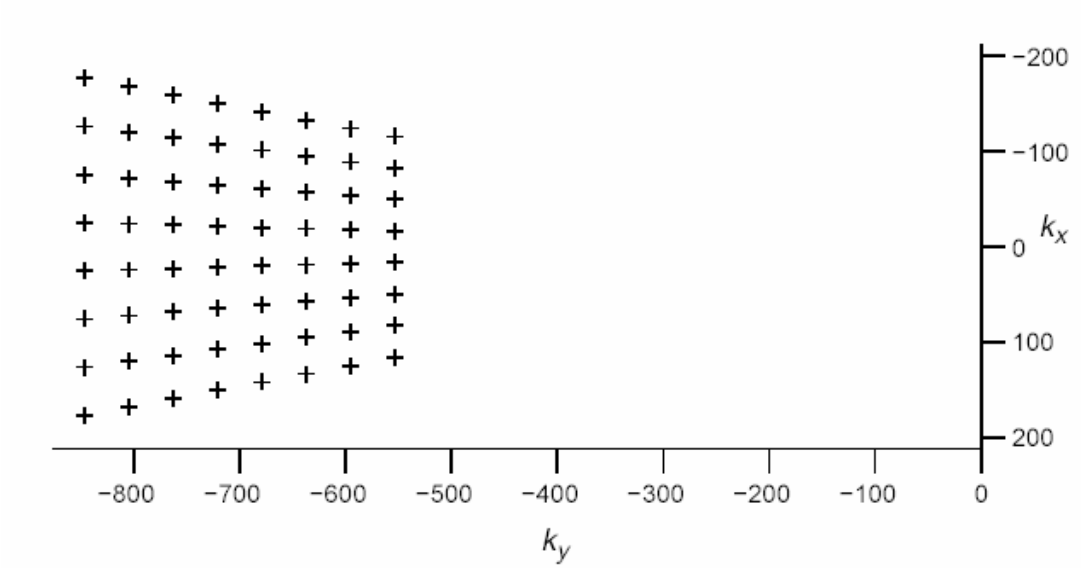

Figure 1: Trapezoidal locations of phase history data in Fourier space resulting from real-time adjustments in radar frequency, chirp rate, and pulse timing.

The data model can be rewritten as

$$X_V(i,n) \approx A_R \sigma(s) \exp j\{k_y(i)s_y\} \cdot \exp j\{k_x(i,n)s_x\}. \quad (4)$$

And will be used in this form for the derivation of the 3 different techniques.

Azimuth Processing using Resampling followed by a FFT

Polar format processing with this method will resample $k_x(i,n)$ in order to remove the dependence upon i. This is accomplished by interpolating such that

$$\left[\omega_o + \gamma_o T_{s,o} i\right] \cdot d\alpha \cdot n = \left[\omega_o\right] \cdot d\alpha \cdot n', \quad (5)$$

or otherwise finding data values at locations

$$n = \frac{\left[\omega_o\right]}{\left[\omega_o + \gamma_o T_{s,o} i\right]} \cdot n' = \left(\frac{1}{1 + \frac{\gamma_o T_{s,o}}{\omega_o} i} \right) \cdot n'. \quad (6)$$

The data are now re-grided to Cartesian locations as pictured in Figure 2.

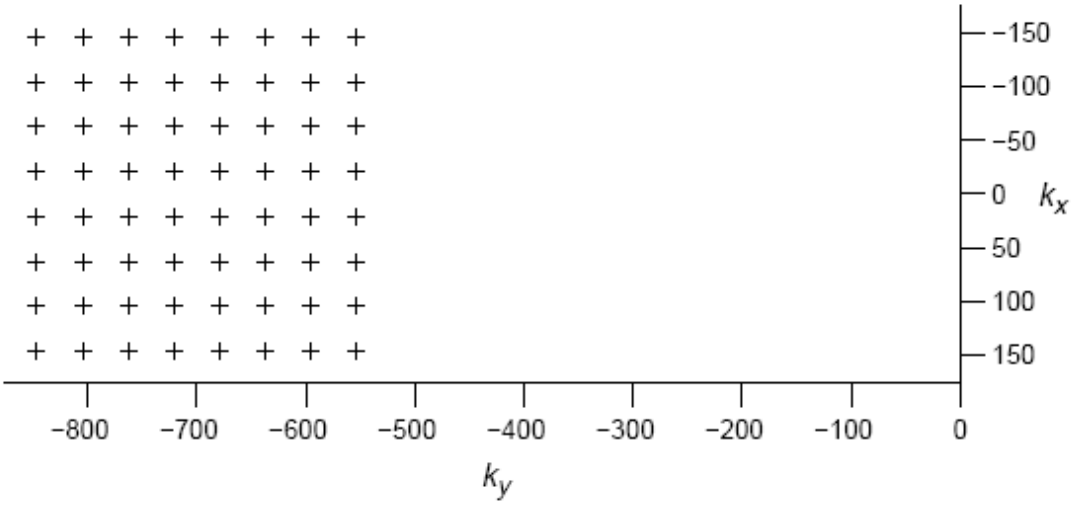

Figure 2: Re-grided Phase History Data Locations

The subsequent 1-D FFT's for image formation will then operate in the *n'* direction and the *i* direction. With a substitution of the exponential with index *n* for the interpolated *n'*, the phase history model has become

$$X_V(i,n') \approx A_R \sigma(s) \exp j\{k_y(i) s_y\} \cdot \exp j\left\{ \frac{2 \cdot \omega_0 \cdot \cos\psi_0}{c} \cdot d\alpha \cdot s_x \cdot n' \right\}. \qquad (7)$$

And the 1-D FFT along the *n'* direction evaluates to

$$FFT_n\left(X_V(i,n')\right) \approx$$

$$\left[A_R \sigma(s) \exp j\{k_y(i) s_y\}\right] \cdot W_{n'}\left(\frac{2 \cdot \omega_0 \cdot \cos\psi_0}{c} \cdot d\alpha \cdot s_x - \frac{2\pi}{N'} u \right) . \qquad (8)$$

where $W_n(\ )$ represents the image 'impulse response' or 'point spread function' in the azimuth direction. We note that in the absence of any window functions or other data tapering

$$W_n(\Omega) = \sum_n e^{j\Omega n} , \qquad (9)$$

which has the shape of a sinc( ) function near its peak value. The resultant FFT then has a peak response

$$\frac{2\pi}{N} u = \frac{2 \cdot \omega_0 \cdot \cos\psi_0}{c} \cdot d\alpha \cdot s_x \qquad (10)$$

that is independent of index *i*. In order to fully optimize the FFT, powers of 2 must be used for the length, and this requires the number of interpolations to be carried out to be the next largest power of 2 greater than the length of the dataset. This will also ensure proper spacing on the output for the samples that will be used in the final image. Note that the number of image pixels is typically much smaller than the total number of azimuth samples.

Azimuth Processing using a FFT Followed by Resampling

Polar format processing with this method will perform a 1-D FFT in the *n* direction such that:

$$FFT_n\left(X_V\left(i,n\right)\right)\approx$$

$$\left[A_R\sigma\left(s\right)\exp j\left\{k_y\left(i\right)s_y\right\}\right]\cdot W_n\left(\frac{2\cdot\omega_0\cdot\cos\psi_0}{c}\cdot d\alpha\cdot s_x\cdot\left(1+\frac{\gamma_0 T_{s,0}}{\omega_0}i\right)-\frac{2\pi}{N}u\right) \quad (11)$$

The resultant sinc( ) function then has a peak response at

$$\frac{2\pi}{N}u=\frac{2\cdot\omega_0\cdot\cos\psi_0}{c}\cdot d\alpha\cdot s_x\cdot\left(1+\frac{\gamma_0 T_{s,0}}{\omega_0}i\right) \quad (12)$$

Note that this response is still dependent on index *i*. To remove this dependence, we can define a new index u' such that

$$\frac{2\pi}{N}u'=\frac{2\cdot\omega_0\cdot\cos\psi_0}{c}\cdot d\alpha\cdot s_x. \quad (13)$$

and resample the data to interpolated positions where

$$u=\frac{\left[\omega_o+\gamma_o T_{s,o}i\right]}{\left[\omega_o\right]}\cdot u'=\left(1+\frac{\gamma_o T_{s,o}}{\omega_o}i\right)\cdot u'. \quad (14)$$

The remaining image formation requires a 1-D FFT in the range direction.

Azimuth Processing using the CZT

Consider an application of the Chirp-Z Transform directly in the *n* direction of the SAR phase history model.

$$CZT_n\left(X_V\left(i,n\right)\right) \approx \left[A_R\sigma\left(s\right)\exp j\left\{k_y\left(i\right)s_y\right\}\right]\cdot$$

$$\sum_n \exp j\cdot\left(\frac{2\cdot\omega_0\cdot\cos\psi_0}{c}\cdot d\alpha\cdot s_x\cdot\left(1+\frac{\gamma_0 T_{s,0}}{\omega_0}i\right)n\right)-j\cdot\Delta_f\cdot u\cdot n \qquad (15)$$

By forcing Δ_f to vary with index *i* such that

$$\Delta_f = \frac{2\pi}{N}\left(1+\frac{\gamma_0 T_{s,0}}{\omega_0}i\right) \qquad (16)$$

The CZT evaluates to

$$CZT_n\left(X_V\left(i,n\right)\right) \approx$$

$$\left[A_R\sigma\left(s\right)\exp j\left\{k_y\left(i\right)s_y\right\}\right]\cdot W_n\left(\left(1+\frac{\gamma_0 T_{s,0}}{\omega_0}i\right)\cdot\left(\frac{2\cdot\omega_0\cdot\cos\psi_0}{c}\cdot d\alpha\cdot s_x-\frac{2\pi}{N}u\right)\right) \qquad (17)$$

This also clearly offers a peak response when

$$\frac{2\pi}{N}u = \frac{2\cdot\omega_0\cdot\cos\psi_0}{c}\cdot d\alpha\cdot s_x \qquad (18)$$

2. Overview & Summary

At this juncture it is important to mention a few of the benefits to each implementation and how it relates to MATLAB as well as some of the shortcomings. The traditional method of polar format has an incredible ease of implementation. Since the Nyquist sampling criteria has been met, the interpolation procedure is simply a weighted sinc function convolution. There are even implementations of this called "fast-sinc interpolators" that remove the transient sine function from the summation. Jackowatz et al. [5] states that a Hann-weighted truncated sinc function with 8 zero crossings and a 1.04 bandwidth reduction factor provides a viable interpolation filter. However, the traditional method of polar format exhibits shortcomings in that first, the interpolation kernel runs relatively slow in a MATLAB environment since it is interpreted; and second, the number of interpolations required is large, considering that the number image pixels will often only be a fraction of the total number of points in the phase history data. Even though this interpolation procedure is easy to implement, it is still often the dominating factor in the processing time required for the algorithm as well as a source of error in the final image.

The second method to be discussed is the post-azimuth transform interpolation. This method has several advantages over the traditional polar format method in that

1. The transform happens prior to interpolation, so it uses the entire frequency space of the data and is not limited to the region of an inscribed rectangle, thereby offering slightly improved resolution.

2. The method does not suffer the increased difficulties in mitigating the problematic sidelobe effects of an exscribed rectangle.

3. Since the transform occurs prior to interpolation, the transform does not use data degraded by inexact interpolations. These interpolation errors are introduced later in the processing chain and lead to an overall lesser error.

4. Since the image usually contains fewer azimuth pixels than the number of original azimuth sample points, fewer interpolations are often required.

5. Along with #3, the method does not require low pass filtering in conjunction with its interpolation as does the traditional method.

6. With the introduction of error later into the polar format processing chain, the post azimuth transform interpolation can use lower-fidelity, smaller kernels than its traditional counterpart for the same image quality (FFT length dependant).

7. As a result of #4 and #6, fewer interpolations with smaller kernels allow greater efficiency, easier implementation, and faster operation.

The major shortcoming of this method is the interpolation error. Inexact interpolations will cause a range direction modulation resulting in false targets within the image. This however can be mitigated by appropriately zero-padding the data to a length beyond the next power of 2. Unfortunately, this also adds computational burden.

The last method is the Chirp Z-Transform image formation algorithm. This method has similar advantages to that of the Post Azimuth Transform Interpolation. These are

1. This technique uses the entire frequency space of the data and is not limited to the region of an inscribed rectangle, thereby offering slightly improved resolution.

2. The method does not suffer the increased difficulties in mitigating the problematic sidelobe effects of an exscribed square.

3. The CZT does not use data degraded by inexact interpolations. In fact, there are no interpolations.

4. The technique does not require low pass filtering of the data set.

5. The CZT can be implemented as a fast convolution using only FFTs.

The CZT shortcoming is that it requires 3 FFTs of 2 times the length of the input sequence. And also 2 complex multiplies of that same length. This can be more inefficient than an interpolation and requires significantly more memory to implement.

3. Detailed Analysis

The first part of this section outlines a change to the existing Polar Format MATLAB implementation utilizing the Chirp-Z Transform that improves performance and memory usage. The second part is the addition of the two other image formation options that perform a more traditional interpolation style image formation.

The original MATLAB implementation was coded by Armin Doerry in [1]. The code, as follows, performs the Chirp Z-Transform method of image formation

MATLAB CZT Image Formation

```
function [cimg, img_parms] = sar_pf(parms,phdat)
% SAR_PF    sar image formation, polar format processing, for MiniSAR

c      = 299.7925e6;    % velocity of propagation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% image formation parameters

rc0    = parms(1);      % nominal range to scene center - m
psi0   = parms(2);      % nominal depression angle - rad
dalpha = parms(3);      % nominal increment in tan(aperture_angle)
N      = parms(4);      % number azimuth samples
w0     = parms(5);      % nominal center frequency - rad/sec
g0     = parms(6);      % nominal chirp rate - rad/sec^2
Ts0    = parms(7);      % nominal A/D sample period - sec
I      = parms(8)       % number range samples
rhox   = parms(9)       % desired x resolution
rhoy   = parms(10);     % desired y resolution
delx   = parms(11)      % desired x pixel spacing
dely   = parms(12);     % desired y pixel spacing
Dx     = parms(13)      % desired x scene diameter
Dy     = parms(14);     % desired y scene diameter
gamma  = parms(15);     % left/right side imaging

[I,N]  = size(phdat);   % redefine N,I based on data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate secondary parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cospsi0    = cos(psi0);              % cosine of depression angle
lambda     = 2*pi*c/w0;              % nominal wavelength
lambda_min = 2*pi*c/(w0+g0*Ts0*I/2); % min wavelength of data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% define window functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
aw_az  = 1.184;
aw_ra  = 1.184;
azwin  = taylor(-35,4,N)  /N ;
rawin  = taylor(-35,4,I).'/I ;
altseq = round(cos([0:pi:(I-1)*pi])).';
rawin  = rawin.*altseq;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% calculate azimuth processing parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delx  = min([ (delx)  (rhox/aw_az) ...
(lambda_min/(2*N*dalpha*cos(psi0)))]);
U     = 2*round(Dx/delx/2)
delx  = Dx/U;                                  % calculate actual delx
rhox  = aw_az*lambda/(2*N*dalpha*cos(psi0)); % actual resolution
os_az = rhox/delx;                             % x oversample factor

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate range processing parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
os_ra = rhoy/dely;                          % desired y oversample factor
I_    = 2*round(I*os_ra/aw_ra/2);           % optimal range fft length
rhoy  = aw_ra*2*pi*c/(2*g0*Ts0*(I )*cos(psi0)); % actual y resolution
dely  = 2*pi*c/(2*g0*Ts0*(I_)*cos(psi0)); % actual y pixel spacing
os_ra = rhoy/dely;    % calculate actual y oversample factor
delr  = dely*cospsi0; % slant-range pixel spacing
rhor  = rhoy*cospsi0; % slant-range resolution

% pick minimum of number of pixels desired and number supported by data
V     = min(2*round(Dy/dely/2),2*round(I*(os_ra/aw_ra)/2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FORM IMAGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% azimuth processing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

argW = -j*2*pi*delx/(N*rhox/aw_az); % nominal angular increment
argA = argW*U/2;                    % nominal initial angle
img1 = j*ones(I,U);


for i = 1:I,

      % Calculate CZT sample spacing modification factor
      beta = 1 + (g0*Ts0/w0)*(i-1-I/2);

      % Perform azimuth CZT
      x    = transpose( phdat(i,:) .* azwin );

      y    = czt(x,U,exp(argW*beta),exp(argA*beta));
      y    = transpose(y);

      img1(i,:) = y * rawin(i);   % apply range window weight and store
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% range processing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% perform range FFT
cimg = fft( img1,I_,1 );
cimg = cimg((I_/2-V/2+1):(I_/2+V/2),:);  % cull meaningful data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% format output image and parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% adjust for left/right side of aircraft
if gamma == 1,
      cimg = fliplr(cimg);
end

img_parms = [ rhox rhoy delx dely ];
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% END
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

This code was benchmarked using a test image and the MATLAB profiler.

MATLAB CZT Image Formation Profiler Results

One way to improve the performance of MATLAB M-files is by using profiling tools. MATLAB provides the M-file Profiler, a graphical user interface that is based on the results returned by the profile function. The above code was run inside the MATLAB environment's Profiler tool to determine where the majority of the processing time was being spent. The results were not surprising in that the CZT function required 44.594 sec of the total 73.29 sec required for the image formation code on the test image.

When the CZT is used as above in image formation, it is used as a spectral zoom. Optimization of the spectral zoom in MATLAB is presented by Martin in [7]. This can improve the performance by 71%. The report also mentions the removal of certain power calculations from the loop. This again increases performance such that the total image formation run time is 17.43 sec for the test image. This is a 76% reduction in the time required to run. The new optimized code is as follows:

MATLAB Optimized CZT Image Formation

```
function [cimg, img_parms] = sar_pf(parms,phdat)
% SAR_PF sar image formation, polar format processing, for MiniSAR

c     = 299.7925e6;     % velocity of propagation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% image formation parameters

rc0    = parms(1);      % nominal range to scene center - m
psi0   = parms(2);      % nominal depression angle - rad
dalpha = parms(3);      % nominal increment in tan(aperture_angle)
N      = parms(4);      % number azimuth samples
w0     = parms(5);      % nominal center frequency - rad/sec
g0     = parms(6);      % nominal chirp rate - rad/sec^2
Ts0    = parms(7);      % nominal A/D sample period - sec
I      = parms(8)       % number range samples
rhox   = parms(9)       % desired x resolution
rhoy   = parms(10);     % desired y resolution
delx   = parms(11)      % desired x pixel spacing
dely   = parms(12);     % desired y pixel spacing
Dx     = parms(13)      % desired x scene diameter
Dy     = parms(14);     % desired y scene diameter
gamma  = parms(15);     % left/right side imaging

[I,N]  = size(phdat);   % redefine N,I based on data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate secondary parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cospsi0     = cos(psi0);              % cosine of depression angle
lambda      = 2*pi*c/w0;              % nominal wavelength
lambda_min  = 2*pi*c/(w0+g0*Ts0*I/2); % min wavelength of data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% define window functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aw_az   = 1.184;
aw_ra   = 1.184;
azwin   = taylor(-35,4,N)  /N ;
rawin   = taylor(-35,4,I).'/I ;
altseq  = round(cos([0:pi:(I-1)*pi])).';
rawin   = rawin.*altseq;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% calculate azimuth processing parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delx  = min([ (delx)  (rhox/aw_az) ...
(lambda_min/(2*N*dalpha*cos(psi0)))]);
U     = 2*round(Dx/delx/2)
delx  = Dx/U;                                   % calculate actual delx
rhox  = aw_az*lambda/(2*N*dalpha*cos(psi0));    % actual resolution
```

```
os_az = rhox/delx;                                  % x oversample factor

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate range processing parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
os_ra = rhoy/dely;                              % desired y oversample factor
I_    = 2*round(I*os_ra/aw_ra/2);               % optimal range fft length
rhoy  = aw_ra*2*pi*c/(2*g0*Ts0*(I )*cos(psi0));  % actual y resolution
dely  = 2*pi*c/(2*g0*Ts0*(I_)*cos(psi0)); % actual y pixel spacing
os_ra = rhoy/dely;     % calculate actual y oversample factor
delr  = dely*cospsi0;  % slant-range pixel spacing
rhor  = rhoy*cospsi0;  % slant-range resolution

% pick minimum of number of pixels desired and number supported by data
V     = min(2*round(Dy/dely/2),2*round(I*(os_ra/aw_ra)/2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FORM IMAGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% azimuth processing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

argW = -j*2*pi*delx/(N*rhox/aw_az); % nominal angular increment
argA = argW*U/2;                    % nominal initial angle
img1 = j*ones(I,U);

nfft = power(2,nextpow2(k+m-1));
kk   = transpose([(-k+1):max(m-1,k-1)]);
kk2  = times(kk,kk)./2;
wPow = times(argW, [kk2]);
nn   = transpose([0:(k-1)]);
aPow = times(-argA , [nn] );

for i = 1:I,

      % Calculate CZT sample spacing modification factor
      beta  = 1 + (g0*Ts0/w0)*(i-1-I/2);

      % Perform azimuth CZT
      x  = transpose( phdat(i,:) .* azwin );

      ww = exp(wPow);
      aa = exp(aPow);

      aa = times(aa , ww(k+nn));
      g  = times(x , aa);

      fy = fft(g,nfft);
      fv = fft(1./[ww(1:(m-1+k))],nfft);
      fy = times(fy,fv);
      y  = ifft(fy);

      y  = times(y(k:(k+m-1)),ww(k:(k+m-1)));
```

```
      y  = transpose(y);

      img1(i,:) = y * rawin(i);    % apply range window weight and store
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% range processing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% perform range FFT
cimg = fft( img1,I_,1 );
cimg = cimg((I_/2-V/2+1):(I_/2+V/2),:);  % cull meaningful data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% format output image and parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% adjust for left/right side of aircraft
if gamma == 1,
      cimg = fliplr(cimg);
end

img_parms = [ rhox rhoy delx dely ];
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% END
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Additional Polar Format Options

The two added options are a Post-Azimuth Transform option and a traditional polar format azimuth interpolation option. These were incorporated with flags at the beginning of the code to denote which option to run. This gives the user a choice of running any of the 3 presented image formation algorithms. The code is as follows

```
function [cimg, img_parms] = sar_pf(parms,phdat)
% SAR_PF sar image formation, polar format processing, for MiniSAR

doCZT = 1;
doTraditionalInterp = 0;
doPostAzFFTInterp = 0;

c     = 299.7925e6;     % velocity of propagation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% image formation parameters

rc0    = parms(1);      % nominal range to scene center - m
psi0   = parms(2);      % nominal depression angle - rad
```

```
dalpha = parms(3);      % nominal increment in tan(aperture_angle)
N      = parms(4);      % number azimuth samples
w0     = parms(5);      % nominal center frequency - rad/sec
g0     = parms(6);      % nominal chirp rate - rad/sec^2
Ts0    = parms(7);      % nominal A/D sample period - sec
I      = parms(8)       % number range samples
rhox   = parms(9)       % desired x resolution
rhoy   = parms(10);     % desired y resolution
delx   = parms(11)      % desired x pixel spacing
dely   = parms(12);     % desired y pixel spacing
Dx     = parms(13)      % desired x scene diameter
Dy     = parms(14);     % desired y scene diameter
gamma  = parms(15);     % left/right side imaging

[I,N]  = size(phdat);   % redefine N,I based on data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate secondary parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cospsi0     = cos(psi0);              % cosine of depression angle
lambda      = 2*pi*c/w0;              % nominal wavelength
lambda_min  = 2*pi*c/(w0+g0*Ts0*I/2); % min wavelength of data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% define window functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aw_az   = 1.184;
aw_ra   = 1.184;
azwin   = taylor(-35,4,N)  /N ;
rawin   = taylor(-35,4,I).'/I ;
altseq  = round(cos([0:pi:(I-1)*pi])).';
rawin   = rawin.*altseq;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% calculate azimuth processing parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delx  = min([ (delx)  (rhox/aw_az) ...
(lambda_min/(2*N*dalpha*cos(psi0)))]);
U     = 2*round(Dx/delx/2)
delx  = Dx/U;                                   % calculate actual delx
rhox  = aw_az*lambda/(2*N*dalpha*cos(psi0));    % actual resolution
os_az = rhox/delx;                              % x oversample factor

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate range processing parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
os_ra = rhoy/dely;                          % desired y oversample factor
I_    = 2*round(I*os_ra/aw_ra/2);           % optimal range fft length
rhoy  = aw_ra*2*pi*c/(2*g0*Ts0*(I )*cos(psi0));  % actual y resolution
dely  = 2*pi*c/(2*g0*Ts0*(I_)*cos(psi0)); % actual y pixel spacing
os_ra = rhoy/dely;     % calculate actual y oversample factor
delr  = dely*cospsi0;  % slant-range pixel spacing
rhor  = rhoy*cospsi0;  % slant-range resolution

% pick minimum of number of pixels desired and number supported by data
V     = min(2*round(Dy/dely/2),2*round(I*(os_ra/aw_ra)/2));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FORM IMAGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% azimuth processing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
argW = -j*2*pi*delx/(N*rhox/aw_az); % nominal angular increment
argA = argW*U/2;                    % nominal initial angle
img1 = j*ones(I,U);

if doCZT
      nfft = power(2,nextpow2(k+m-1));
      kk   = transpose([(-k+1):max(m-1,k-1)]);
      kk2  = times(kk,kk)./2;
      wPow = times(argW, [kk2]);
      nn   = transpose([0:(k-1)]);
      aPow = times(-argA , [nn] );
end

for i = 1:I,

      % Calculate CZT sample spacing modification factor
      beta  = 1 + (g0*Ts0/w0)*(i-1-I/2);

      % Perform azimuth CZT
      x  = transpose( phdat(i,:) .* azwin );

      if doCZT
            ww = exp(wPow);
            aa = exp(aPow);

            aa = times(aa , ww(k+nn));
            g  = times(x , aa);

            fy = fft(g,nfft);
            fv = fft(1./[ww(1:(m-1+k))],nfft);
            fy = times(fy,fv);
            y  = ifft(fy);

            y  = times(y(k:(k+m-1)),ww(k:(k+m-1)));
            y  = transpose(y);

      end

      if doTraditionalInterp
            nfft = 2^(nextpow2(N)+1);
            yi   = sincinterp(x.'.*(exp(argA*beta.*-[0:N-1])),...
                  [0:N-1],[0:nfft-1]*(N*rhox) / ...
                  (beta*delx*aw_az*nfft),16,'hann');

            y1   = fft(yi,nfft);

            % cull data and scale for equivalence
            y    = y1(1:U)*N*rhox/(beta*delx*aw_az*nfft);
```

```
            img1(i,:) = y * rawin(i);
      end

      if doPostAzFFTInterp
            nfft = ceil(2^(nextpow2(N)+1));
            yfft = fft(x.'.*(exp(argA*beta.*-[0:N-1])),nfft);
            y    = sincinterp(yfft, [0:nfft-1],[0:U-1]*beta*...
                   delx*aw_az/(N*rhox)*nfft,16,'hann');
      end

      img1(i,:) = y * rawin(i);    % apply range window weight and store
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% range processing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% perform range FFT
cimg = fft( img1,I_,1 );
cimg = cimg((I_/2-V/2+1):(I_/2+V/2),:);  % cull meaningful data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% format output image and parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% adjust for left/right side of aircraft
if gamma == 1,
      cimg = fliplr(cimg);
end

img_parms = [ rhox rhoy delx dely ];
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% END
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The above code simply adds flags at the beginning which determine the method of image formation to run. The two added interpolation methods both use a 16 tap, Hanning weighted sinc interpolator. Other options are possible for interpolation such as linear, cubic or spline, and those can easily be implemented by replacing the sincinterp line of code. For example, preliminary exploration with a simple linear interpolator was performed with the post azimuth transform option. When the raw phase history data is zero-padded to 4 times the length of the next power of 2 for the original sequence, the error between linear interpolation and the CZT was found to be visibly incomprehensible. In fact, the measured impulse response is a fraction of a pixel wider for the linear interpolated case. The image in Figure 3 below is a 4-in resolution image taken of the VEE range and formed with the CZT method. Figure 4 is the same raw phase history data, but formed with the linear interpolation scheme described above. The impulse responses for both are in Figures 5 and 6 respectively.

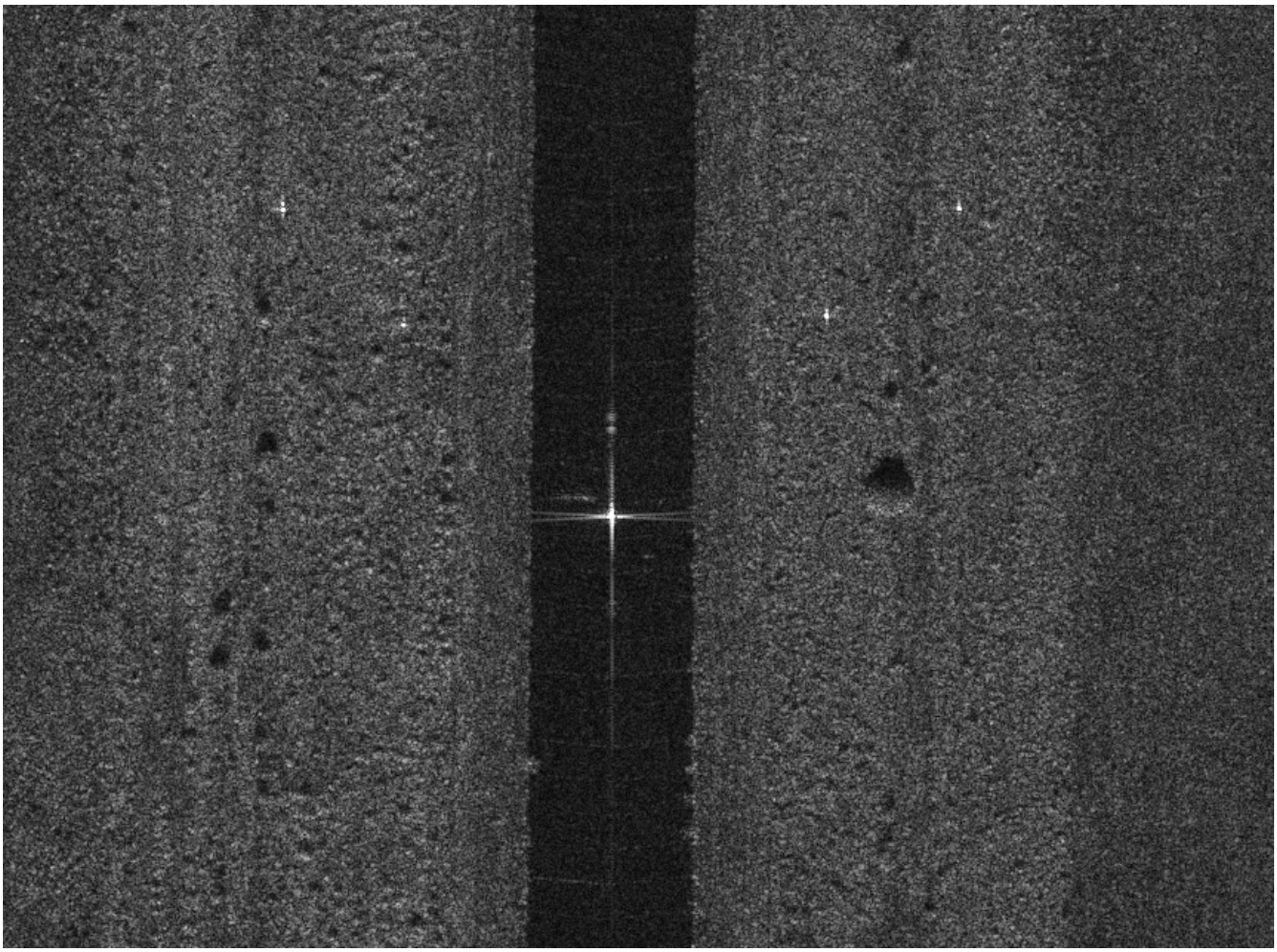

Figure 3: VEE range, 4in Resolution, CZT Image Formation Option

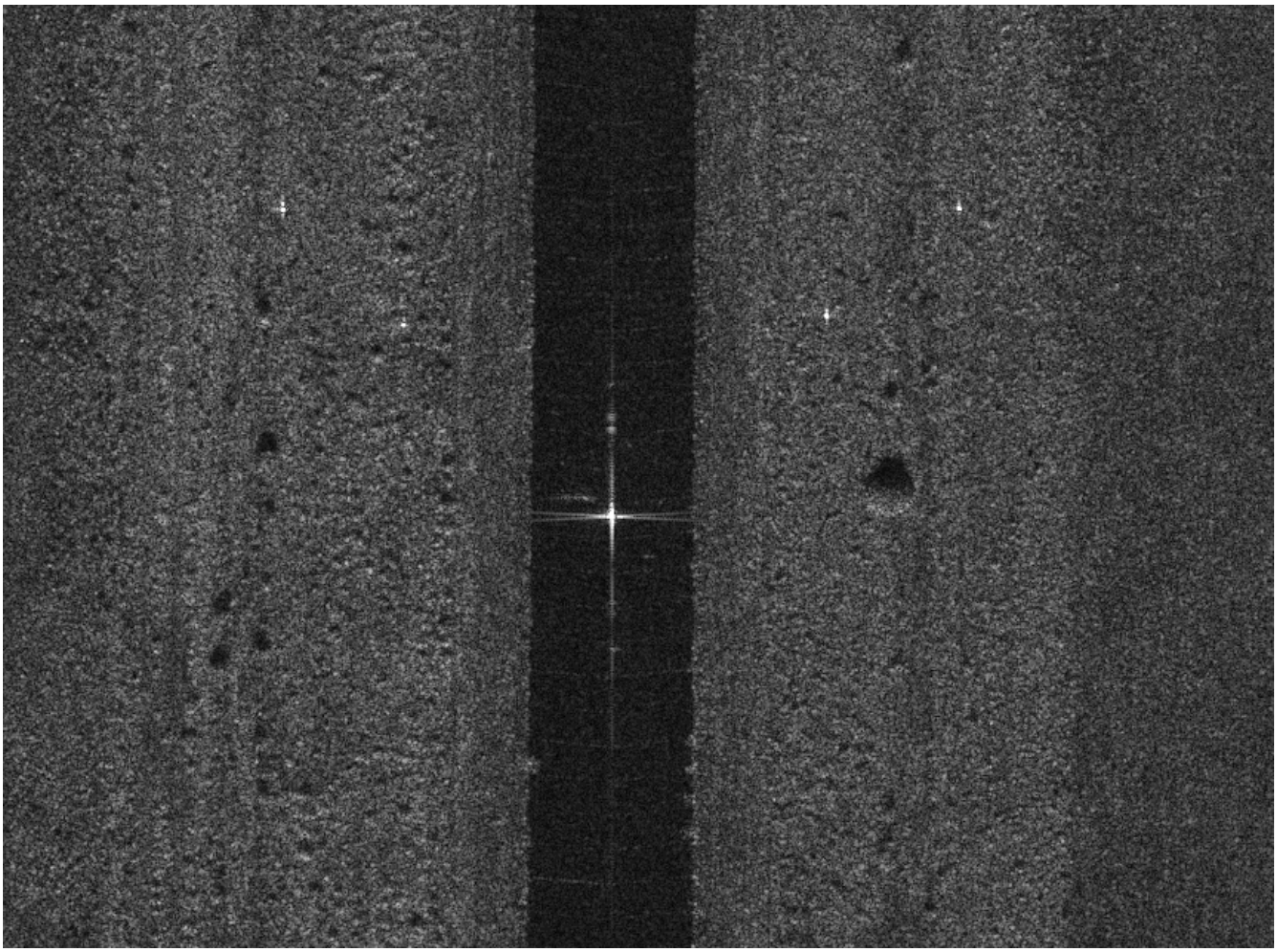

Figure 4: VEE range, 4in Resolution, Linear Interpolation Post Azimuth Transform Image Formation Option

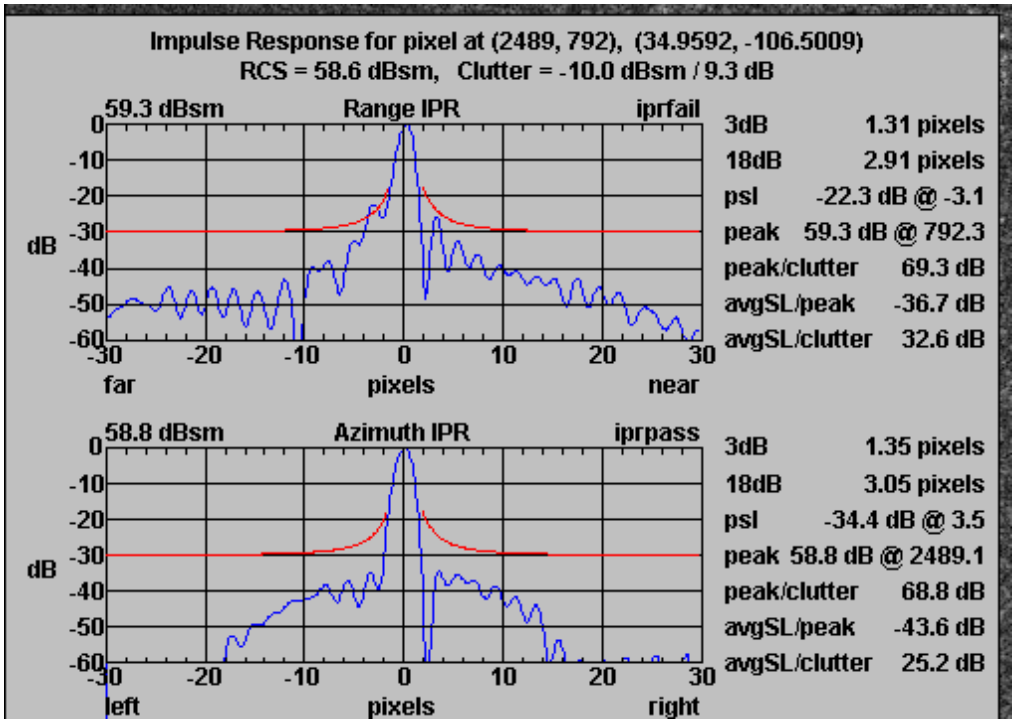

Figure 5: Impulse Response of Peak in CZT Formed Image

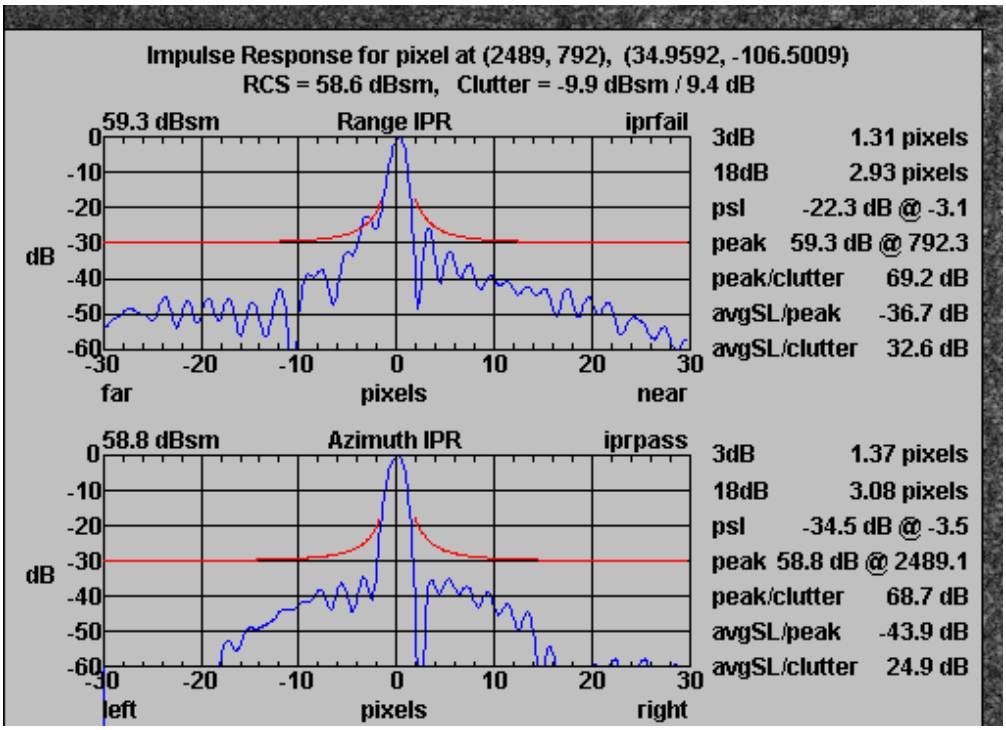

Figure 6: Impulse Response of Peak in Post Azimuth Linear Interpolated Image

These impulse responses show that linear interpolation is not only possible for image formation with a post azimuth transform option, but when compared to the CZT formed image, the 3dB width is only .02 pixels wider for the linear interpolated image. The normalized phase error between the two methods is also small, but more conclusive experiments need to be run to determine the full extent of the difference and whether or not there are effects with interferometric processing or coherent change detection.

4. Conclusions

The first change to the existing MATLAB Polar Format is an optimization of the Chirp Z-Transform that improves performance and memory usage. The changed algorithm allows MATLAB to be used for image formation applications where it was previously not possible. In fact, the test case forms and displays an image 76% faster than the original code, where a large portion (1/3) of the processing time is now devoted to reading the raw phase history data from disk. The optimized code is also capable of forming images on a Windows platform and has demonstrated the formation of an 8k x 2k raw phase history image (4 byte complex samples) with only 1 Gigabyte of RAM. This was formerly an impossibility due to memory limitations, but now, this type of large aperture formation is possible with the Windows based MATLAB.

The second change is the addition of two new possible image formation options that perform a more traditional interpolation style image formation. These options allow the continued exploration of possible interpolation methods for image formation. Some preliminary work has been done to show that image error between the CZT method and either interpolation method, as presented with the sinc interpolator, is negligible. Furthermore, preliminary results also show that image formation can be accomplished with a simple interpolator and the post azimuth transform method. A full exploration of possible interpolation effects with a linear interpolator or other methods still needs to be conducted, but early results are promising in that the phase error is small between the two methods.

References

[1] Armin W. Doerry, “Real-time Polar-Format Processing for Sandia’s Testbed Radar Systems”, Sandia National Laboratories Report SAND2001-1644P, June 2001.

[2] Brown, W.M., and R.J. Fredericks, “Range-Doppler Imaging with Motion though Resolution Cells,” IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-5, No. 1, January 1969, pp 98-102.

[3] Walker, J.L., “Range-Doppler Imaging of Rotating Objects,” IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 1, January 1980, pp. 23-52.

[4] Grant Martin, Armin Doerry, and Michael Holzrichter, “A Novel Polar Format Algorithm for SAR Images Utilizing Post Azimuth Transform Interpolation” Sandia National Laboratories Report SAND2005-5510, September 2005.

[5] C. V. Jakowatz Jr., D. E. Wahl, P. H. Eichel, D. C. Ghiglia, P. A. Thompson, *Spotlight-Mode Synthetic Aperture Radar: A Signal Processing Approach*, ISBN 0-7923-9677-4, Kluwer Academic Publishers, 1996.

[6] Walter G. Carrara, Ron S. Goodman, Ronald M. Majewski, *Spotlight Synthetic Aperture Radar, Signal Processing Algorithms*, ISBN 0-89006-728-7, Artech House Publishers, 1995.

[7] Grant Martin, “Chirp Z-Transform Spectral Zoom Optimization with MATLAB” Sandia National Laboratories Report SAND2005-7084, November 2005.

DISTRIBUTION

Unlimited Release

1	MS 0509	M. W. Callahan	5300
1	MS 0529	B. L. Remund	5340
1	MS 0519	B. L. Burns	5340
1	MS 0503	T. J. Mirabal	5341
1	MS 0519	W. H. Hensley	5342
1	MS 0519	T. P. Bielek	5342
1	MS 1330	A. W. Doerry	5342
1	MS 0519	D. Harmony	5342
1	MS 0519	J. A. Hollowell	5342
1	MS 0529	S. S. Kawka	5342
1	MS 0519	M. S. Murray	5342
1	MS 0519	B. G. Rush	5342
1	MS 0519	D. G. Thompson	5342
1	MS 1330	K. W. Sorensen	5345
1	MS 1330	J. A. Bach	5345
1	MS 0529	B. C. Brock	5345
1	MS 1330	D. F. Dubbert	5345
1	MS 1330	G. R. Sloan	5345
1	MS 1330	S. M. Becker	5348
1	MS 0519	S. M. Devonshire	5348
1	MS 1330	M. W. Holzrichter	5348
1	MS 1330	O. M. Jeromin	5348
1	MS 1330	G. D. Martin	5348
1	MS 1330	P. G. Ortiz	5348
1	MS 1330	D. C. Sprauer	5348
1	MS 1330	A. D. Sweet	5348
1	MS 0519	M. E. Thompson	5348
1	MS 0519	L. M. Wells	5354
1	MS 0519	D. L. Bickel	5354
1	MS 0519	J. T. Cordaro	5354
1	MS 0519	J. DeLaurentis	5354
1	MS 1330	A. E. Mihalik	5354
1	MS 0311	F. M. Dickey	2616
2	MS 0899	Technical Library	4536 1
2	MS 9018	Central Technical Files	8945-1

Sandia National Laboratories